ECE 6332 Design Review2: Register File Design Optimization with Virtual Prototyping Tool (ViPro)

Group member:

Ningxi Liu (ECE6332) Email:nl6cg@virginia.edu

Shawn Chen (ECE6332) Email:sc7cq@virginia.edu

1. Publication Review

The register file cells are similar to SRAM cells, so a lot of problems are common in these two kinds of memories. For example, the read and write abilities are conflicted with each other. Koichi Takeda proposed a multi-step WL control scheme[1] to realize read assist and write assist at the same time, as Figure 1 left shows. The idea is really smart because it takes the BL voltage into consideration to calculate the actual read margin, which becomes larger with the decreasing of BL voltage. Another interesting point is shown in the right part, which tells us that the composition of delay is different under different bank architecture and assist methods. This is really an interesting research question for Vipro, because Vipro are trying to give the energy and delay results for different architecture of register file banks. It will also be useful if Vipro can separately give delay and energy data of each part, then we can figure out what's the most effective assist method for any given bank structure.

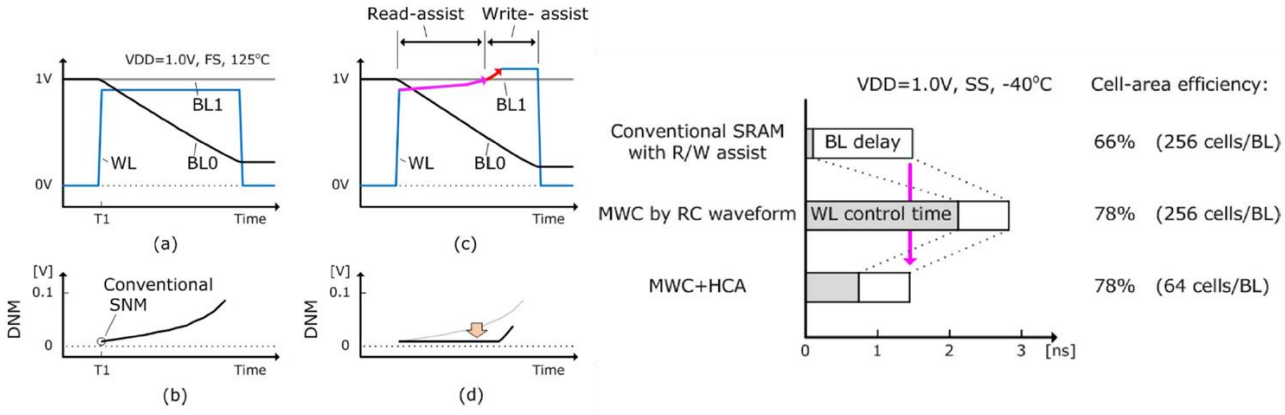


Figure 1 Multi step WL control scheme(left); Delay time composition under different assists (right).

Atsushi Kawasumi gives the idea that for SRAM, low voltage is not always efficient in reducing dynamic power[2]. The first reason is dynamic power consumed by SRAM is decided by 1/2*Vbl*Cbl*VDD, which is linearly reduced with VDD shrinking, not like the dynamic power of logic circuits which have a quadratic relation with VDD. The second reason is shown in Figure 2, that the variation worsens the access time

severely in low voltage, because the delay distribution of BL is much wider in low voltages. As the consequence, faster BLs should wait for the worst case BL so that the whole array can function right, so the extra energy is consumed. This paper supports the idea that SA is not needed in low voltage applications, for the reason that the speed improvement resulted from SA is really trivial compared to the worst case delay.

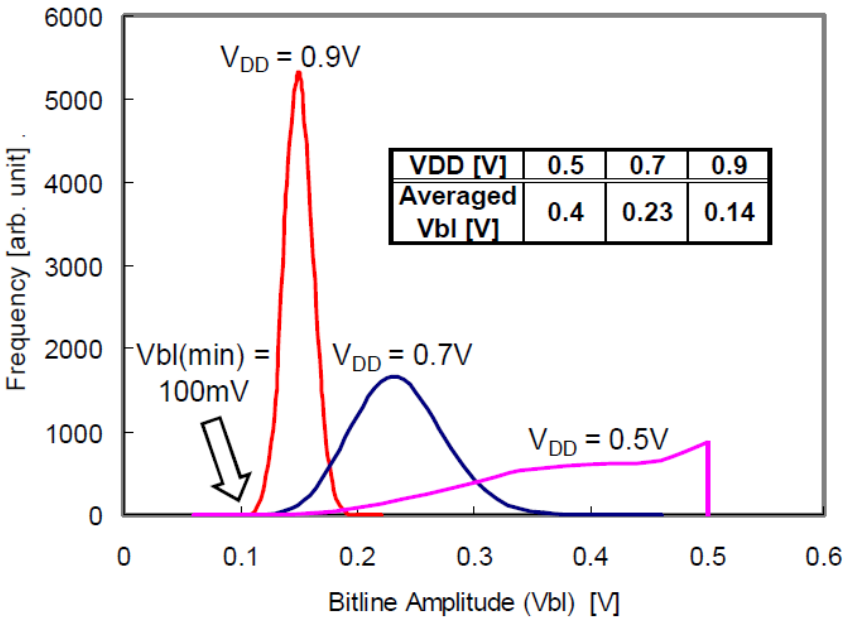


Figure 2 Bitline amplitude distribution under different voltages

2. Works have been done

2.1 Comparison between Vipro and spice netlist

2.1.1 Goal

The research question we want to answer by this work is that, can an optimization tool like ViPro generate power and delay estimates for a RF that approach the accuracy of SPICE, and how to verify the model of RF in ViPro?

To answer the above research question, we have to obtain the simulation results from a spice netlist. For that purpose, we should be able to generate a complete register file schematic which can match one of the register file Vipro cases.

2.1.2 Building complete register file schematic with SKILL

To build a more flexible register file schematic, we choose to use SKILL to generate any schematic we want according to some given parameters for the register file bank. In the current version, the SKILL

script is able to automatically produce a register file bank with multiple bitcells and peripheral circuits. The parameters are:

- NRow, which stands for the number of rows in a bank.
- NCol_Mux, which stands for the number of columns that share a SA.
- Wordsize, which means the number of bits can be read or write at one operation.
- bcType, which can be used to choose different types of register file cells.

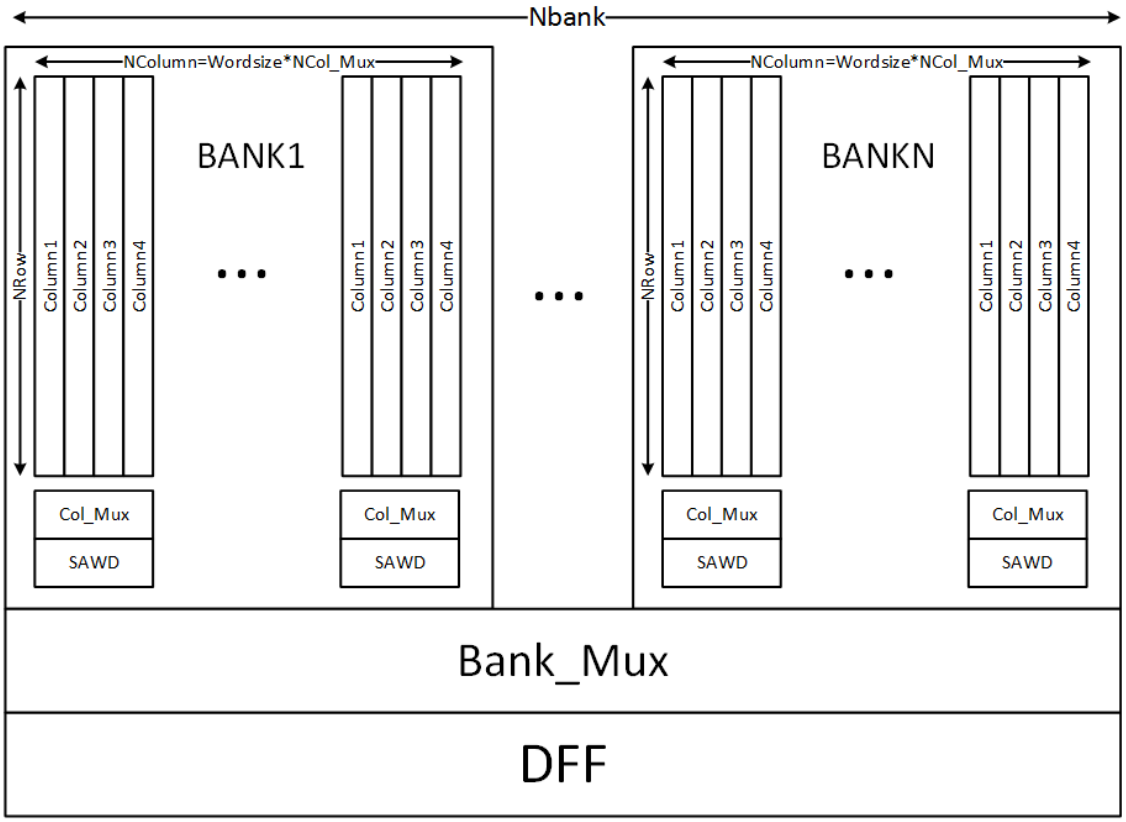


Figure 3 Hierarchy of register file banks

Figure 3 shows the overview of the organization of register file banks. There are two kinds of bank in the schematic implementation, which are the active bank and unselected banks. The active bank has more complicated input stimuli and relatively smaller sub circuits to reduce simulation time. The unselected banks are used to provide standby power composition for the complete register schematic, to be mentioned, the standby leakage often can be neglected during active period compared to active power.

There are also some wire parasitic capacitance models in the generated schematic to realize high simulation precision. The wire capacitance models are abstracted as following ways:

- For BL & WL:
  CBL=NRow*Cbl
  CWL=Ncol_Mux*Wordsize*Cwl
- For inputs cap of SAE WEN PCH DIN CSEL BankSEL:

C=NColMux*Wordsize*Cwl

- For BankMux output cap:
  C=Nbank*NColMux*Wordsize*Cwl
- For DFF output cap:
  C=Nbank/2*NColMux*Wordsize*Cwl

2.1.3 Simulation results comparison

We generated five different register bank structures, the total memory sizes of which are all 16384 bits. Figure 4 plots the data of write & read delay, and write & read energy consumption of results from RF schematic and Vipro. As we can see from the figure, the delay data of RF schematic is overall larger than the Vipro data, and energy data is very close in several items.

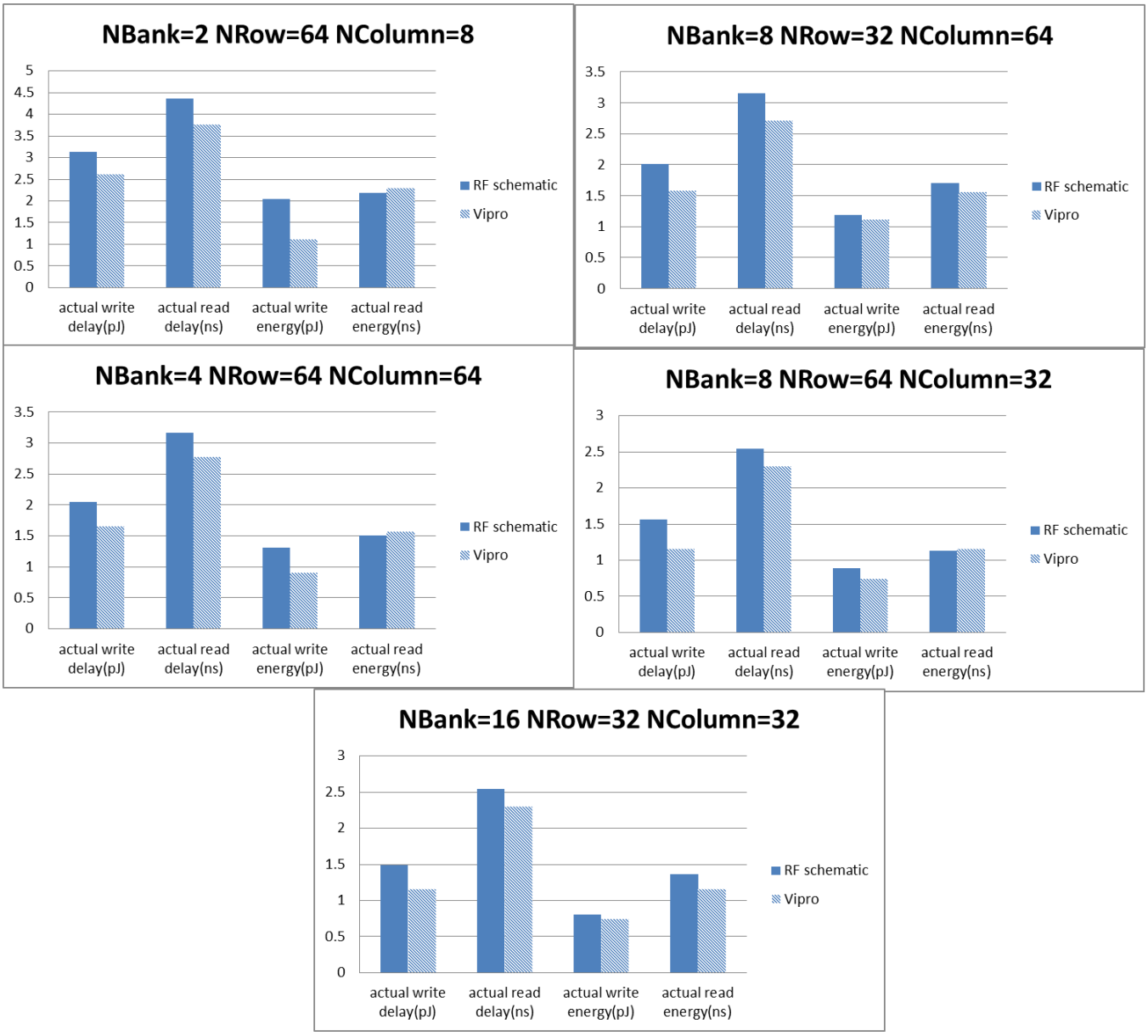


Figure 4 Simulation results comparison between different hierarchy structures of Vipro and RF schematic

From Figure 5, we can see that there exist differences between the results of Vipro and RF schematic. The differences of read delay and energy don't have a large variation, which means there may be some systematic discrepancy between Vipro and RF schematic, and the trends of data generated by Vipro are definitely right. For write operation, the large difference partly resulted from the absolute value is smaller. In general, the results of Vipro are reliable.

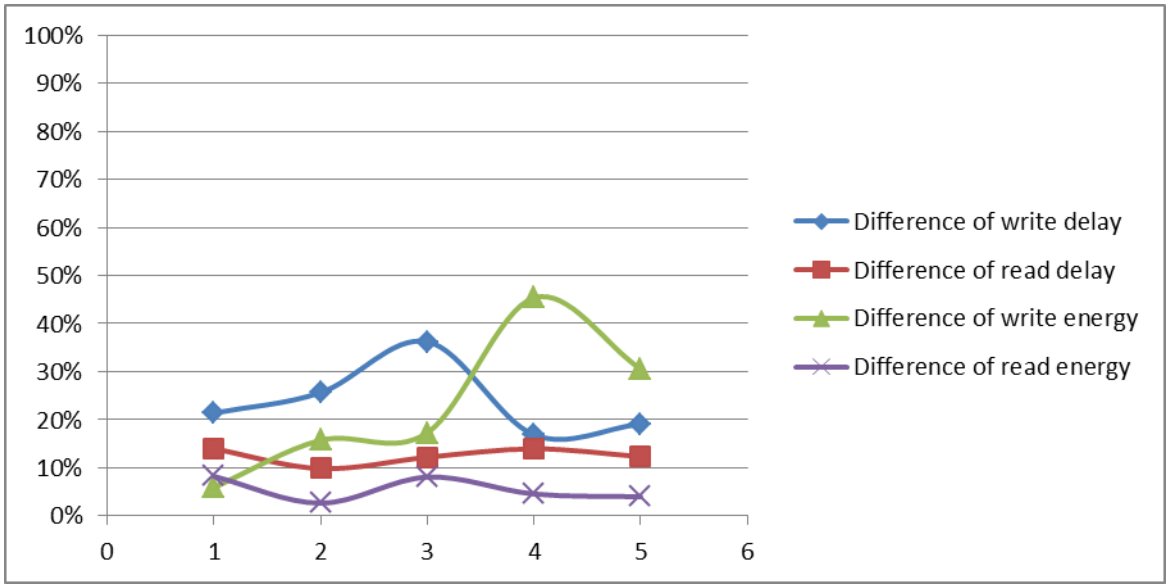
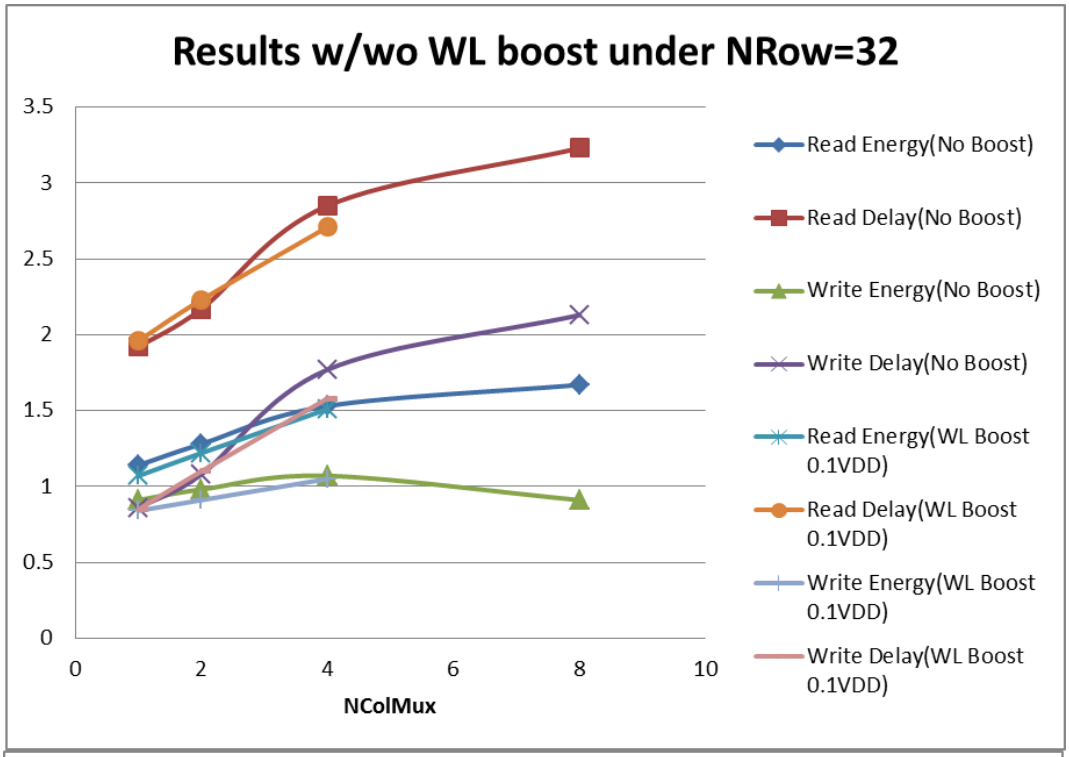


Figure 5 Difference of simulation results for read & write delay and read & write energy between RF banks

2.2 Register file performance with and without WL boost

WL boost is a popular design metric in register file design. Figure 6 Simulation results w/wo WL boost under NRow=32(Up), Simulation results w/wo WL boost under NRow=64(Down) Illustrates the influence of WL boost on read & write energy and read & write delay. As we can see from the figure, both read and write delay is improved after given 0.1 VDD WL boost. To make things better, the read and write energy after WL boost actually shrinks a little, which is resulted from the operation time decreasing. To do this research in the future, we can give more results of different WL boost amplitude and under different number of rows.



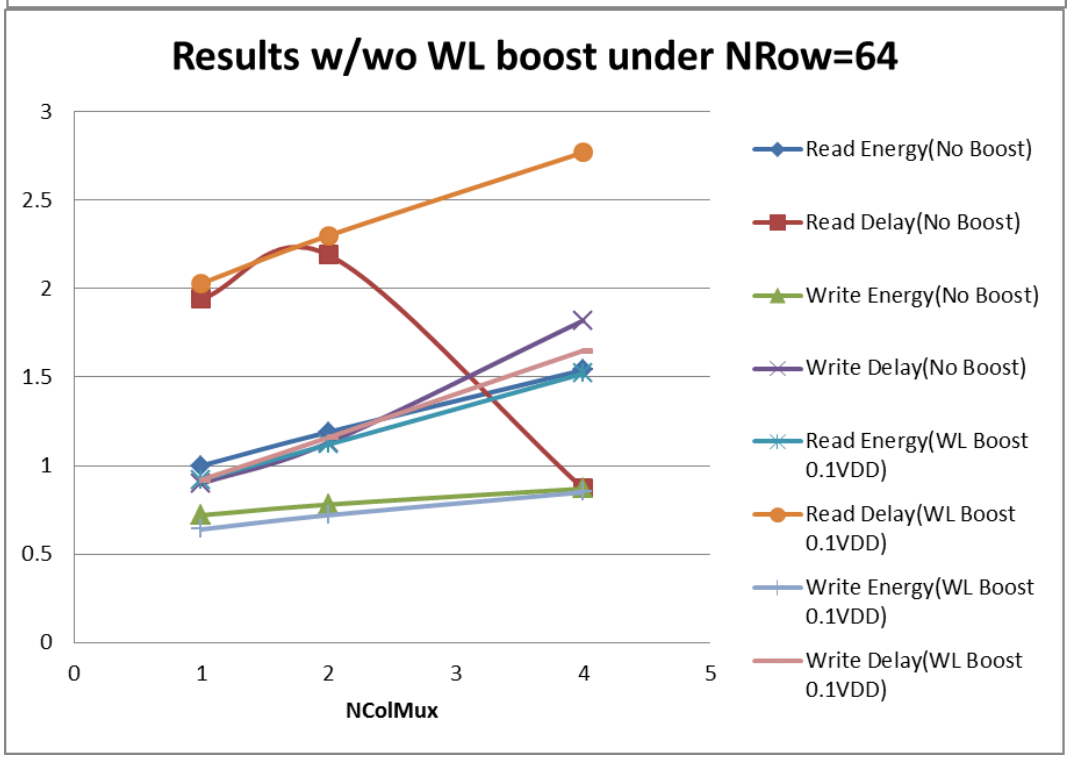


Figure 6 Simulation results w/wo WL boost under NRow=32(Up), Simulation results w/wo WL boost under NRow=64(Down)

2.3 Vipro code improvement

During developing Vipro for register file, many of our efforts have been spent in making the code to work, which including the cpp code and some TASE tests failing issue.

Original ViPro source code is written in only one C++ file and it didn't use header files, which makes it hard to extend. As we hope to make the code more well-structured and easier for future developers to maintain, we have re-constructed the ViPro source code. Following are the major modifications.

2.3.1 Separated classes.

The original ViPro source code is divided into 5 main classes, RegFile, calculator, parser, userInput, periphery. RegFile is corresponding to the RegFile model we've designing. It has all the necessary parameters stored as private properties. It also contains senseAmp, bitCell, rowDecoder, columnMux, timingBlock, DFF and bankMux, which are sub-blocks of RegFile. Calculator is the class that contains iteration executor and simulation handler. Parser is used for parsing and extracting user configuration from user.m. UserInput is responsible for interaction with the user and storing intermediate results. Periphery is where the sub-blocks are defined and implemented.

2.3.2 Fine granularity and extendable user-defined knobs

ViPro basically reads the user.m configuration and parses it to extract the knobs. However, the extracted knobs are only a small set which are hard coded inside ViPro source code. Considering our research questions that promises to search the effective knobs contributing to RegFile design, ViPro should provide the user with more featured user-defined knobs. Thus we add extra parsing logic so that ViPro can support new syntax in user.m. As an example, we would like to explore how RegFile performs when Vdd goes down but original ViPro doesn't support user-defined Vdd. So we add this into ViPro to overwrite the template.

2.3.3 Bug elimination

We've been trying to simulate the RegFile structure both in ViPro and SKILL. While increasing the bankMux number in SKILL, this parameter in ViPro seems unchanged in output. After putting much effort debugging, we found that original ViPro confines the maximum bankMux to be 16. To avoid similar hard coded bugs, we removed related hard coded lines and started to use variables.

2.3.4 Multiple level verbose debug log

This is one of our important progress that helps much with our project. An implementation of 3-level verbose debug log allows us to obtain enough information for debugging. Setting up DEBUG flag turns the level 1 log on. Level 1 log gives a birdview of progress of whole simulation task. DEBUG2 turns level 2 on, which gives details of each function in execution. DEBUG3 turns level 3 on, which gives the most detailed information including variable value in each iteration of each loop.

2.4 Speeding Vipro up by reducing unnecessary knobs combination

According one of our research question, reducing the undesired iteration is important. It can save a large amount of computation power from being wasted. In other words, how to speed up ViPro by changing the value of knobs?

We've found that some combinations of NBANKS, NCOLS and NROWS don't work due to violation of total memory size. Thus, we can simply do a boundary check and skip the violated iterations instead of waiting for failure report from ocean. Another approach is to give the range of each knob as well as the excluded specific value. ViPro will check the current value before simulation. If current value fits in one of the excluded value, this iteration will be terminated and going to next one directly.

Now we are not able to provide the expected excluded value list. But we will use some statistical techniques to find those values.

3. Work to do

3.1 To recognize the most significant components of energy and delay

The purpose of this work is to figure out which accounts for the largest part of energy and delay under different register file bank architecture. For example, for register bank with larger number of row, the main factor that increasing the delay maybe the BL delay, while for a bank with longer WL, the cause may be different.

As a result, we want Vipro to be capable of simulating each sub circuit in given combination of design knobs such as NRow and NBank. And the output of each simulation should be in an organized way, so that further analysis can easily continue.

3.2 To enable Vipro in low voltage application

Currently Vipro can't generate data under lower voltages like 0.7V, because some TASE tests are designed only can meet the requirement of high VDD. However, low voltage is quite important for low power design, so Vipro should be able to support low voltage simulation. After the improvement, we can use Vipro to compare active power and standby power ratio under low voltage, and the results can help decide which bank architecture is eligible for low voltage operation.

3.3 To realize sweep of parameters in Vipro

In details, we want Vipro to support sweeping those input stimuli during simulation, such as VDD, WL_Offset, SA_Offset and so on. Also, the sweep of these tests should be consistent with the design knobs chosen. After realizing this function, the performance of a bank architecture under different voltages can be researched, which is very useful in register file design.

References:

[1] Koichi Takeda. Multi-Step Word-Line Control Technology in Hierarchical Cell Architecture for Scaled-Down High-Density SRAMs. IEEE JOURNAL OF SOLID-STATE CIRCUITS, VOL. 46, NO. 4, APRIL 2011.

[2] Energy Efficiency Degradation Caused by Random Variation in Low-Voltage SRAM and 26% Energy Reduction by Bitline Amplitude Limiting (BAL) Scheme. IEEE Asian Solid-State Circuits Conference, November 14-16, 2011 / Jeju, Korea.